STAT 139: Final Project

Danny Kim, Christopher Lee, Karina Wang, Daniel Son

2022-12-14

EDA

```r
set.seed(139)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Load team data
team_data = list()
team_wins <- list()
drop = c("W", "L")
for (year in 1997:2022) {
  df1 = read.csv(paste("data/teams_data/batting", year, ".csv", sep=""))
  df2 = read.csv(paste("data/teams_data/pitching", year, ".csv", sep=""))
  df3 = read.csv(paste("data/teams_data/fielding", year, ".csv", sep=""))
  df_tot = merge(merge(df1, df2, by="Tm", suffixes=c(".bat", ".pitch")), df3, by="Tm", suffixes=c("", "

  df_tot = df_tot[
    !(df_tot$Tm %in% c("", "League Average")),
    !(names(df_tot) %in% drop)
  ]
  df_tot$Tm = factor(df_tot$Tm)
  team_data[[year]] = df_tot
  team_wins[[year]] = df_tot[, c("Tm", "W.L.")]
}

# Load player data
years <- 1997:2022
bps <- c("batting", "pitching", "fielding")
player_data <- list()
```

```r
for (year in years) {
  player_data[[year]] <- list()
  for (bp in bps) {
    player_data[[year]][[bp]] <- read.csv(paste("data/player_data/", bp, year, ".csv", sep=""))
    quant_cols <- names(select_if(player_data[[year]][[bp]], is.numeric))
    for (col in quant_cols) {
      # impute data with mean
      df <- player_data[[year]][[bp]]
      player_data[[year]][[bp]][is.na(player_data[[year]][[bp]][,col]),col] <- mean(df[,col], na.rm=TRUE
    }
  }
}

fa_data = list()
for (year in years) {
  fa_data[[year]] = read.csv(paste("data/fa_data/fa", year, ".csv", sep=""))
  fa_data[[year]]$WAR3[is.na(fa_data[[year]]$WAR3)] = 0
}
```

```r
set.seed(139)
# Data Cleaning for the Team Data
team_wins <- list()
for (year in years) {
  team_wins[[year]] <- team_data[[year]][!(team_data[[year]]$Tm %in% c("", "League Average")), c("Tm",
}
```

```r
set.seed(139)
# Clean player data
for (year in years) {
  for (bp in bps) {
    player_data[[year]][[bp]]$year <- year
    player_data[[year]][[bp]]$year_adj <- year - 1997
  }
}

for (year in years) {
  player_data[[year]][["pitching"]] = player_data[[year]][["pitching"]][!is.infinite(player_data[[year]]
}

long_team_names <- team_data[[year]][!(team_data[[year]]$Tm %in% c("", "League Average")),]$Tm
short_team_names <- c("ARI", "ATL", "BAL", "BOS", "CHC", "CHW", "CIN", "CLE", "COL", "DET",
                      "HOU", "KCR", "LAA", "LAD", "MIA", "MIL", "MIN", "NYM", "NYY", "OAK",
                      "PHI", "PIT", "SDP", "SFG", "SEA", "STL", "TBR", "TEX", "TOR", "WSN")
agg_data <- list()
for (year in years) {
  agg_data[[year]] <- list()
  for (bp in bps) {
    quant_cols <- names(select_if(player_data[[year]][[bp]], is.numeric))
    agg_data[[year]][[bp]] <- player_data[[year]][[bp]][, c("Tm", quant_cols)] %>%
      group_by(Tm) %>%
      summarise(across(quant_cols, ~weighted.mean(., w = G)))
    agg_data[[year]][[bp]] <- agg_data[[year]][[bp]][!(agg_data[[year]][[bp]]$Tm == "TOT"),]
    agg_data[[year]][[bp]]$long_Tm <- factor(
```

```
      agg_data[[year]][[bp]]$Tm,
      levels=short_team_names,
      labels=long_team_names
    )
  }
}
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(quant_cols)
##
##   # Now:
##   data %>% select(all_of(quant_cols))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

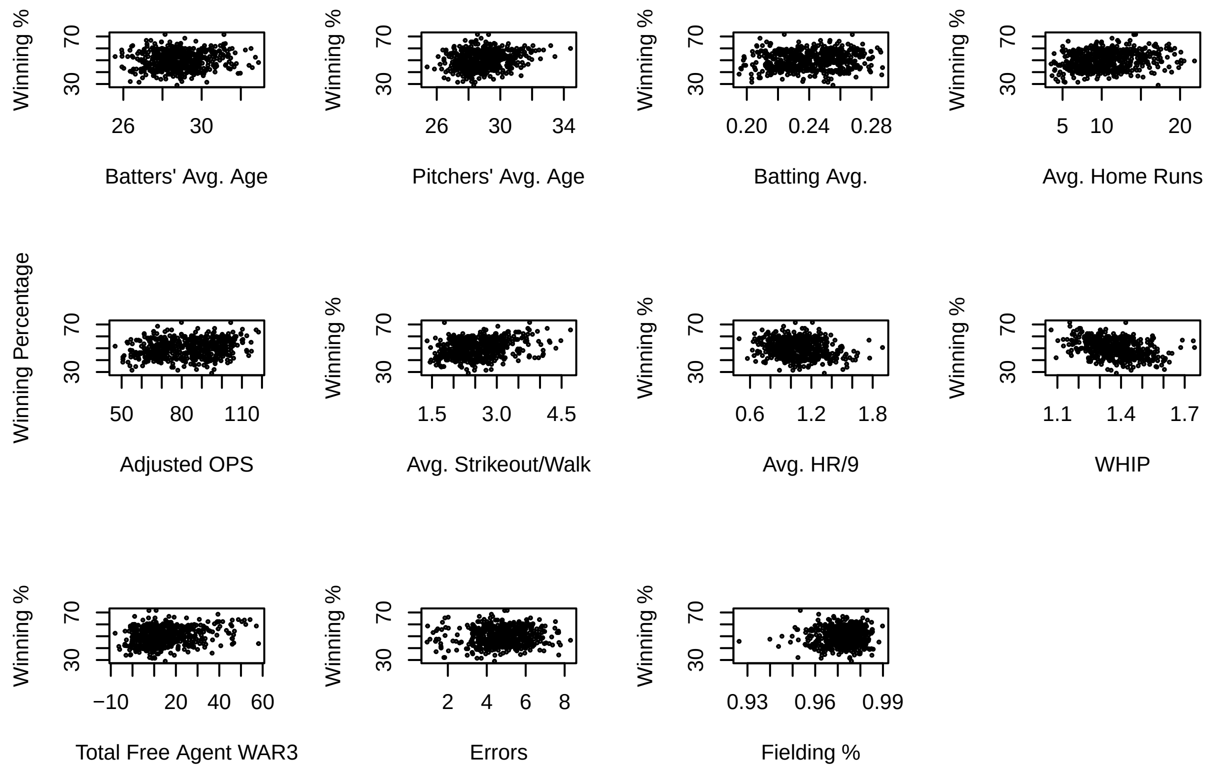
```
player_combo <- list()
for (year in years) {
  player_combo[[year]] <- merge(merge(agg_data[[year]][[bps[1]]], agg_data[[year]][[bps[2]]], by="Tm", s
}

agg_fa_data <- list()
for (year in years) {
  agg_fa_data[[year]] = fa_data[[year]] %>% group_by(To.Team) %>% summarise(tot_fa_war3=sum(WAR3), num_f
}

# add response variable to player data
player_with_wins <- list()
for (year in 1997:2021) {
  player_with_wins[[year]] <- merge(player_combo[[year]], team_wins[[year+1]], by.x="long_Tm.pitch", by
}

player_with_wins_fa <- list()
for (year in 1997:2021) {
  player_with_wins_fa[[year]] <- merge(player_with_wins[[year]], agg_fa_data[[year]], by.x="long_Tm.pitc
}


player_with_wins_combined = bind_rows(player_with_wins_fa, )
player_with_wins_combined$W.L..same_year = 100 * player_with_wins_combined$W.L..same_year
player_with_wins_combined$W.L..next_year = 100 * player_with_wins_combined$W.L..next_year

drop_cols = c("long_Tm.pitch", "Rk.bat", "G.bat", "long_Tm.bat", "Rk.pitch", "W", "L", "G.pitch", "long_
              "Age", "GS", "CG", "GS.field", "CG.field", "Rdrs", "Rdrs.yr", "Rgood")
player_with_wins_combined = player_with_wins_combined[, !(names(player_with_wins_combined) %in% drop_col

n.rows = nrow(player_with_wins_combined)
n.train = 0.8 * n.rows
train.rows = sample(n.rows, n.train)
train.df = player_with_wins_combined[train.rows,]
colnames(train.df)[colnames(train.df) == 'OPS.'] <- 'OPSplus'
colnames(train.df)[colnames(train.df) == 'ERA.'] <- 'ERAplus'
```

```r
test.df = player_with_wins_combined[-train.rows,]
colnames(test.df)[colnames(test.df) == 'OPS.'] <- 'OPSplus'
colnames(test.df)[colnames(test.df) == 'ERA.'] <- 'ERAplus'
```

```r
set.seed(139)
# train.df
names(train.df)
```

```
##  [1] "Tm"               "Age.bat"          "PA"               "AB"
##  [5] "R.bat"            "H.bat"            "X2B"              "X3B"
##  [9] "HR.bat"           "RBI"              "SB"               "CS"
## [13] "BB.bat"           "SO.bat"           "BA"               "OBP"
## [17] "SLG"              "OPS"              "OPSplus"          "TB"
## [21] "GDP"              "HBP.bat"          "SH"               "SF"
## [25] "IBB.bat"          "year.bat"         "year_adj.bat"     "Age.pitch"
## [29] "W.L..same_year"   "ERA"              "GF"               "SHO"
## [33] "SV"               "IP"               "H.pitch"          "R.pitch"
## [37] "ER"               "HR.pitch"         "BB.pitch"         "IBB.pitch"
## [41] "SO.pitch"         "HBP.pitch"        "BK"               "WP"
## [45] "BF"               "ERAplus"          "FIP"              "WHIP"
## [49] "H9"               "HR9"              "BB9"              "SO9"
## [53] "SO.W"             "year.pitch"       "year_adj.pitch"   "Rk"
## [57] "G"                "Inn"              "Ch"               "PO"
## [61] "A"                "E"                "DP"               "Fld."
## [65] "Rtot"             "Rtot.yr"          "RF.9"             "RF.G"
## [69] "year"             "year_adj"         "W.L..next_year"   "tot_fa_war3"
## [73] "num_fas"
```

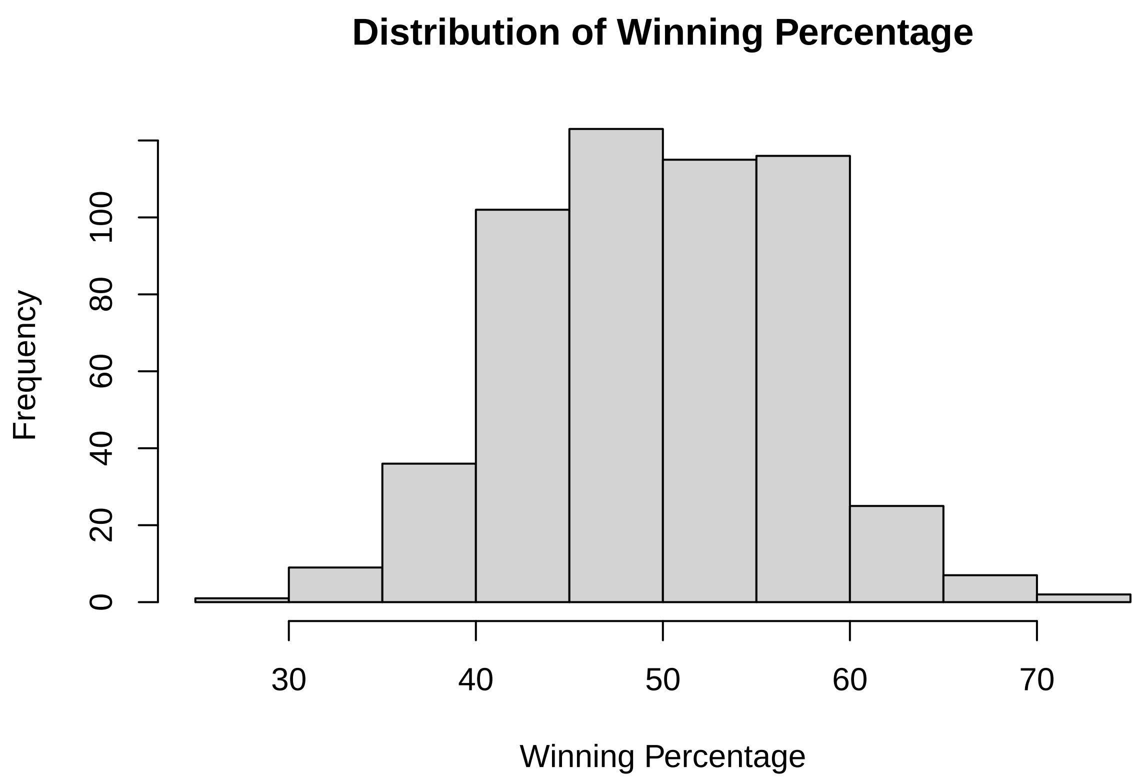
```r
set.seed(139)
# Explore Potential Predictors
par(mfrow=c(3,4))
plot(W.L..next_year ~ Age.bat, data=train.df,
     xlab="Batters' Avg. Age", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ Age.pitch, data=train.df,
     xlab="Pitchers' Avg. Age", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ BA, data=train.df,
     xlab="Batting Avg.", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ HR.bat, data=train.df,
     xlab="Avg. Home Runs", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ OPSplus, data=train.df,
     xlab="Adjusted OPS", ylab="Winning Percentage", cex=0.3)
plot(W.L..next_year ~ SO.W, data=train.df,
     xlab="Avg. Strikeout/Walk", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ HR9, data=train.df,
     xlab="Avg. HR/9", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ WHIP, data=train.df,
     xlab="WHIP", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ tot_fa_war3, data=train.df,
     xlab="Total Free Agent WAR3", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ E, data=train.df,
     xlab="Errors", ylab="Winning %", cex=0.3)
plot(W.L..next_year ~ Fld., data=train.df,
     xlab="Fielding %", ylab="Winning %", cex=0.3)
```




```r
set.seed(139)
# Summary statistics for winpct
summary(train.df$W.L..next_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   29.00   44.25   50.00   50.00   55.60   71.70
```

```r
# Histogram for winpct
hist(train.df$W.L..next_year, main="Distribution of Winning Percentage",
     xlab="Winning Percentage")
```




```
set.seed(139)
# Correlation matrix
cor(train.df[, c("W.L..next_year", "Age.bat", "Age.pitch", "BA", "HR.bat", "OPS", "SO.W", "HR9", "WHIP"
```

```
##                W.L..next_year     Age.bat   Age.pitch          BA      HR.bat
## W.L..next_year     1.00000000  0.08224671  0.21742466  0.12566525  0.22847988
## Age.bat            0.08224671  1.00000000  0.55001609  0.17936468  0.14955980
## Age.pitch          0.21742466  0.55001609  1.00000000  0.11001110  0.17415837
## BA                 0.12566525  0.17936468  0.11001110  1.00000000  0.54086115
## HR.bat             0.22847988  0.14955980  0.17415837  0.54086115  1.00000000
## OPS                0.21118275  0.14622701  0.15726077  0.91088907  0.70048120
## SO.W               0.27971182 -0.12666156  0.01433416 -0.16523565  0.04242539
## HR9               -0.21914037 -0.15698253 -0.11519862  0.06053036  0.13528353
## WHIP              -0.37861475  0.01837121 -0.09029580  0.15570078 -0.02679298
## tot_fa_war3        0.24658861  0.20214373  0.27480015  0.08147753  0.12028178
## E                  0.05870997  0.07619294  0.08515064  0.15724569  0.24875661
## Fld.               0.07749318  0.11616465  0.15603131  0.03284303  0.09552827
##                          OPS        SO.W         HR9        WHIP tot_fa_war3
## W.L..next_year  0.2111827540  0.27971182 -0.21914037 -0.37861475  0.24658861
## Age.bat         0.1462270149 -0.12666156 -0.15698253  0.01837121  0.20214373
## Age.pitch       0.1572607748  0.01433416 -0.11519862 -0.09029580  0.27480015
## BA              0.9108890698 -0.16523565  0.06053036  0.15570078  0.08147753
## HR.bat          0.7004811952  0.04242539  0.13528353 -0.02679298  0.12028178
## OPS             1.0000000000 -0.03748966  0.20036799  0.10150780  0.14490028
## SO.W           -0.0374896628  1.00000000 -0.08710935 -0.74611516  0.11303284
```

```
## HR9             0.2003679869 -0.08710935  1.00000000  0.44954337 -0.01632062
## WHIP            0.1015078012 -0.74611516  0.44954337  1.00000000 -0.07786594
## tot_fa_war3     0.1449002814  0.11303284 -0.01632062 -0.07786594  1.00000000
## E               0.0483299012 -0.34676528 -0.17488952  0.19344202 -0.07855023
## Fld.            0.0001700022  0.03183827 -0.13485425 -0.15172872  0.03494714
##                          E          Fld.
## W.L..next_year  0.05870997  0.0774931799
## Age.bat         0.07619294  0.1161646523
## Age.pitch       0.08515064  0.1560313061
## BA              0.15724569  0.0328430314
## HR.bat          0.24875661  0.0955282742
## OPS             0.04832990  0.0001700022
## SO.W           -0.34676528  0.0318382734
## HR9            -0.17488952 -0.1348542488
## WHIP            0.19344202 -0.1517287249
## tot_fa_war3    -0.07855023  0.0349471418
## E               1.00000000 -0.0300911417
## Fld.           -0.03009114  1.0000000000
```

```
cor(train.df[, c("W.L..next_year", "Age.bat", "Age.pitch", "BA", "HR.bat", "OPS", "SO.W", "HR9", "WHIP"
```

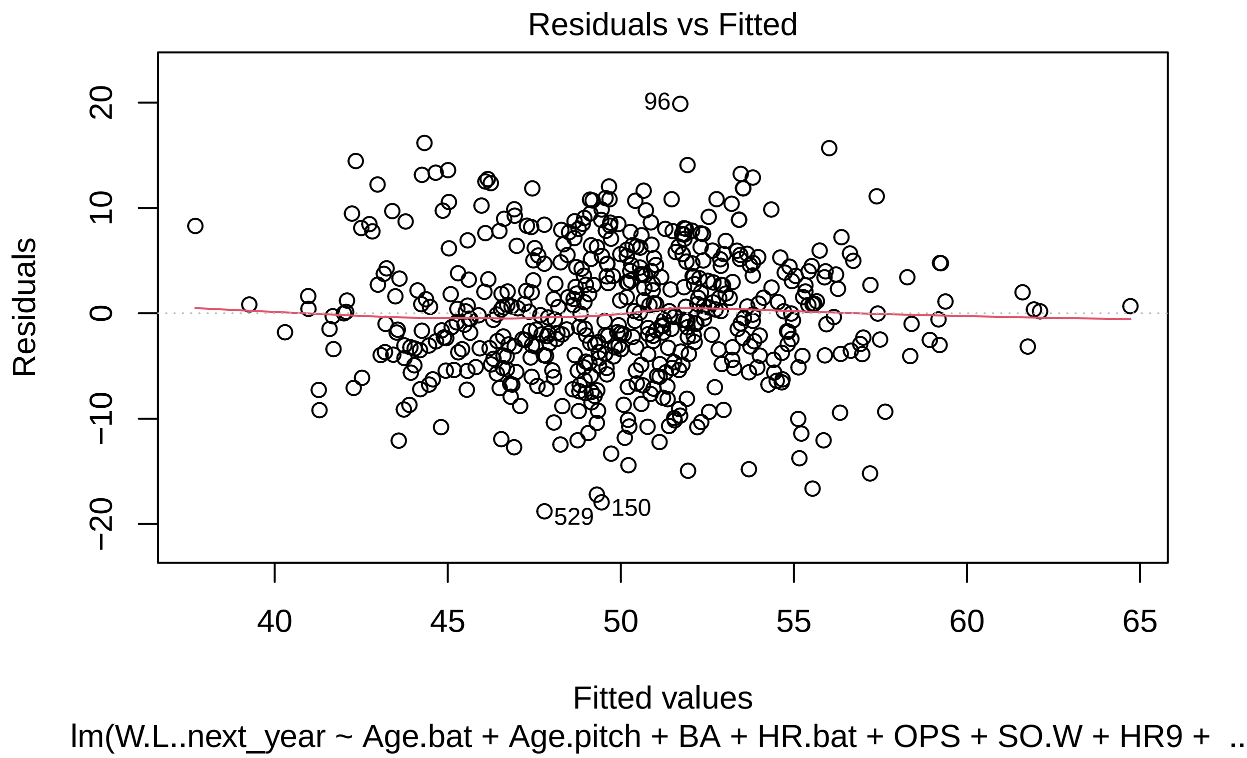
```
##                W.L..next_year      Age.bat    Age.pitch          BA
## W.L..next_year    1.000000000 0.0067645221 0.0472734818 0.015791755
## Age.bat           0.006764522 1.0000000000 0.3025177042 0.032171688
## Age.pitch         0.047273482 0.3025177042 1.0000000000 0.012102441
## BA                0.015791755 0.0321716877 0.0121024412 1.000000000
## HR.bat            0.052203057 0.0223681334 0.0303311363 0.292530787
## OPS               0.044598156 0.0213823399 0.0247309513 0.829718897
## SO.W              0.078238703 0.0160431499 0.0002054681 0.027302819
## HR9               0.048022500 0.0246435148 0.0132707225 0.003663925
## WHIP              0.143349129 0.0003375014 0.0081533311 0.024242733
## tot_fa_war3       0.060805942 0.0408620872 0.0755151242 0.006638587
## E                 0.003446860 0.0058053636 0.0072506310 0.024726208
## Fld.              0.006005193 0.0134942264 0.0243457685 0.001078665
##                      HR.bat          OPS         SO.W          HR9         WHIP
## W.L..next_year 0.0522030566 4.459816e-02 0.0782387030 0.0480224998 0.1433491285
## Age.bat        0.0223681334 2.138234e-02 0.0160431499 0.0246435148 0.0003375014
## Age.pitch      0.0303311363 2.473095e-02 0.0002054681 0.0132707225 0.0081533311
## BA             0.2925307865 8.297189e-01 0.0273028187 0.0036639247 0.0242427326
## HR.bat         1.0000000000 4.906739e-01 0.0017999134 0.0183016333 0.0007178638
## OPS            0.4906739048 1.000000e+00 0.0014054748 0.0401473302 0.0103038337
## SO.W           0.0017999134 1.405475e-03 1.0000000000 0.0075880387 0.5566878383
## HR9            0.0183016333 4.014733e-02 0.0075880387 1.0000000000 0.2020892372
## WHIP           0.0007178638 1.030383e-02 0.5566878383 0.2020892372 1.0000000000
## tot_fa_war3    0.0144677066 2.099609e-02 0.0127764222 0.0002663625 0.0060631047
## E              0.0618798534 2.335779e-03 0.1202461613 0.0305863439 0.0374198160
## Fld.           0.0091256512 2.890074e-08 0.0010136757 0.0181856684 0.0230216060
##                  tot_fa_war3            E         Fld.
## W.L..next_year 0.0608059421 0.0034468602 6.005193e-03
## Age.bat        0.0408620872 0.0058053636 1.349423e-02
## Age.pitch      0.0755151242 0.0072506310 2.434577e-02
## BA             0.0066385874 0.0247262081 1.078665e-03
## HR.bat         0.0144677066 0.0618798534 9.125651e-03
## OPS            0.0209960915 0.0023357794 2.890074e-08
```

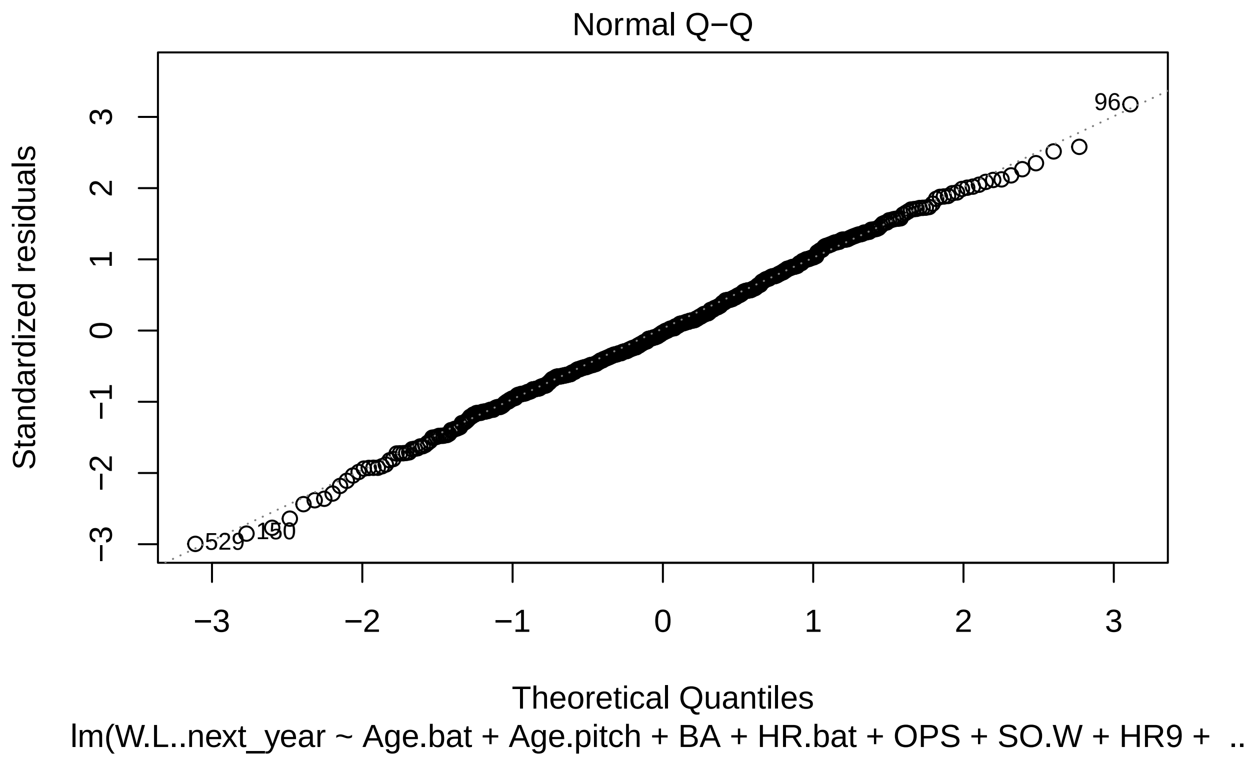
```
## SO.W           0.0127764222 0.1202461613 1.013676e-03
## HR9            0.0002663625 0.0305863439 1.818567e-02
## WHIP           0.0060631047 0.0374198160 2.302161e-02
## tot_fa_war3    1.0000000000 0.0061701382 1.221303e-03
## E              0.0061701382 1.0000000000 9.054768e-04
## Fld.           0.0012213027 0.0009054768 1.000000e+00
```

```
set.seed(139)
# Baseline Multiple Regression Model
baseline <- lm(W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat +
                 OPS + SO.W + HR9 + WHIP + tot_fa_war3 + E + Fld. , data = train.df)
summary(baseline)
```

```
##
## Call:
## lm(formula = W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat +
##     OPS + SO.W + HR9 + WHIP + tot_fa_war3 + E + Fld., data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.7940  -3.9996  -0.1834   4.3942  19.8820
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   28.6771    38.5617   0.744 0.457410
## Age.bat       -0.2027     0.2738  -0.740 0.459511
## Age.pitch      0.4866     0.2695   1.805 0.071580 .
## BA          -167.7755    43.1161  -3.891 0.000113 ***
## HR.bat        -0.1662     0.1318  -1.261 0.207948
## OPS           86.8924    17.1237   5.074 5.41e-07 ***
## SO.W           0.4083     0.8394   0.486 0.626876
## HR9           -5.0925     1.7943  -2.838 0.004714 **
## WHIP         -21.0898     4.8067  -4.388 1.39e-05 ***
## tot_fa_war3    0.1011     0.0249   4.061 5.63e-05 ***
## E              1.0228     0.2805   3.646 0.000293 ***
## Fld.          22.9958    37.3691   0.615 0.538577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.345 on 524 degrees of freedom
## Multiple R-squared:  0.2933, Adjusted R-squared:  0.2784
## F-statistic: 19.77 on 11 and 524 DF,  p-value: < 2.2e-16
```

```
set.seed(139)
# Asssess Linear Model Assumptions
plot(baseline, which=c(1,2))
```


Residuals vs Fitted
Residuals
Fitted values
lm(W.L..next_year ~ Age.bat + Age.pitch + BA + HR.bat + OPS + SO.W + HR9 + ..
96
529
150
20
10
0
−10
−20
40
45
50
55
60
65




```
set.seed(139)
RMSE <- function(y,yhat){
  SSE = sum((y-yhat)^2)
  SST = sum((y - mean(y))^2)
  return(sqrt(SSE/length(y)))
}

R2 <- function(y,yhat) {
  SSE = sum((y-yhat)^2)
  SST = sum((y-mean(y))^2)
  r.squared <- 1 - (SSE / SST)
  return(r.squared)
}
baseline.trainRMSE = RMSE(train.df$W.L..next_year, predict(baseline, newdata=train.df))
baseline.testRMSE = RMSE(test.df$W.L..next_year, predict(baseline, newdata=test.df))
baseline.trainR2 = R2(train.df$W.L..next_year, predict(baseline, newdata=train.df))
baseline.testR2 = R2(test.df$W.L..next_year, predict(baseline, newdata=test.df))
```

Linear Regression

```
set.seed(139)
colnames(train.df)
```

```
##   [1] "Tm"                "Age.bat"          "PA"                "AB"
```

```
##  [5] "R.bat"          "H.bat"          "X2B"            "X3B"
##  [9] "HR.bat"         "RBI"            "SB"             "CS"
## [13] "BB.bat"         "SO.bat"         "BA"             "OBP"
## [17] "SLG"            "OPS"            "OPSplus"        "TB"
## [21] "GDP"            "HBP.bat"        "SH"             "SF"
## [25] "IBB.bat"        "year.bat"       "year_adj.bat"   "Age.pitch"
## [29] "W.L..same_year" "ERA"            "GF"             "SHO"
## [33] "SV"             "IP"             "H.pitch"        "R.pitch"
## [37] "ER"             "HR.pitch"       "BB.pitch"       "IBB.pitch"
## [41] "SO.pitch"       "HBP.pitch"      "BK"             "WP"
## [45] "BF"             "ERAplus"        "FIP"            "WHIP"
## [49] "H9"             "HR9"            "BB9"            "SO9"
## [53] "SO.W"           "year.pitch"     "year_adj.pitch" "Rk"
## [57] "G"              "Inn"            "Ch"             "PO"
## [61] "A"              "E"              "DP"             "Fld."
## [65] "Rtot"           "Rtot.yr"        "RF.9"           "RF.G"
## [69] "year"           "year_adj"       "W.L..next_year" "tot_fa_war3"
## [73] "num_fas"
```

```r
set.seed(139)
# full linear regression models

# ignore Rk.bat, R.bat, RBI, year.bat, year_adj.bat, W, L, R.pitch, year.pitch
# year_adj.pitch, Rk, WL..next_year, year, year_adj, ERA, ERAplus
# Rtot, Rtot.yr, Rdrs, Rgood (hard to interpret)
lm.full <- lm(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B +
                HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                GF + SHO + SV + IP + H.pitch + HR.pitch +
                BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                G + Inn + Ch + PO + A + E + DP + Fld. +
                RF.9 + RF.G + tot_fa_war3 + num_fas,
              data = train.df)
lmfull.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.full, newdata=train.df))
```

```
## Warning in predict.lm(lm.full, newdata = train.df): prediction from a rank-
## deficient fit may be misleading
```

```r
lmfull.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.full, newdata=test.df))
```

```
## Warning in predict.lm(lm.full, newdata = test.df): prediction from a rank-
## deficient fit may be misleading
```

```r
lmfull.trainR2 = R2(train.df$W.L..next_year, predict(lm.full, newdata=train.df))
```

```
## Warning in predict.lm(lm.full, newdata = train.df): prediction from a rank-
## deficient fit may be misleading
```

```r
lmfull.testR2 = R2(test.df$W.L..next_year, predict(lm.full, newdata=test.df))
```

```
## Warning in predict.lm(lm.full, newdata = test.df): prediction from a rank-
## deficient fit may be misleading
```

```
lm.fullinteraction <- lm(W.L..next_year ~ (Age.bat + PA + AB + H.bat + X2B + X3B +
                HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                GF + SHO + SV + IP + H.pitch + HR.pitch +
                BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                G + Inn + Ch + PO + A + E + DP + Fld. +
                RF.9 + RF.G + tot_fa_war3 + num_fas)^2, data = train.df)
lmfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.fullinteraction, newdata=train.d
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = train.df): prediction from a
## rank-deficient fit may be misleading
```

```
lmfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.fullinteraction, newdata=test.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = test.df): prediction from a
## rank-deficient fit may be misleading
```

```
lmfullinteraction.trainR2 = R2(train.df$W.L..next_year, predict(lm.fullinteraction, newdata=train.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = train.df): prediction from a
## rank-deficient fit may be misleading
```

```
lmfullinteraction.testR2 = R2(test.df$W.L..next_year, predict(lm.fullinteraction, newdata=test.df))
```

```
## Warning in predict.lm(lm.fullinteraction, newdata = test.df): prediction from a
## rank-deficient fit may be misleading
```

```
set.seed(139)
# Ridge Regression

library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked _by_ '.GlobalEnv':
##
##     R2, RMSE
```

```r
# regularize full model
X.full = model.matrix(lm.full)[,-1] # drop intercept
best_lambda = cv.glmnet(X.full, train.df$W.L..next_year, alpha=0, lambda=10^seq(-4, 4, 0.1))$lambda.min
```

```
## [1] 1.258925
```

```r
ridges.full = glmnet(X.full, train.df$W.L..next_year, alpha=0,
                     lambda=best_lambda)
imp <- as.data.frame(varImp(ridges.full, lambda=best_lambda))
imp <- data.frame(overall = imp$Overall,
          names = rownames(imp))
imp[order(imp$overall,decreasing = T),][1:10,]
```

```
##      overall names
## 13 38.477736   OBP
## 52 18.456584  Fld.
## 14 15.601157   SLG
## 15 10.724433   OPS
## 39  6.301111  WHIP
## 26  4.141528   SHO
## 54  2.620403  RF.G
## 35  2.396358    BK
## 41  1.381744   HR9
## 40  1.274424    H9
```

```r
X.full.test = model.matrix(lm.full, data=test.df)[,-1] # drop intercept

yhats.full.train = predict(ridges.full, X.full)
ridgesfull.trainRMSE = RMSE(train.df$W.L..next_year, yhats.full.train) # train RMSE
ridgesfull.trainR2 = R2(train.df$W.L..next_year, yhats.full.train) # train R2

yhats.full.test = predict(ridges.full, X.full.test)
#plot(RMSE.ridges.full.test~log(ridges.full$lambda, 10), type='l')
ridgesfull.testRMSE = RMSE(test.df$W.L..next_year, yhats.full.test) # test RMSE
ridgesfull.testR2 = R2(test.df$W.L..next_year, yhats.full.test) # test R2
```

```r
set.seed(139)
# regularize full interaction model
X.fullinteraction = model.matrix(lm.fullinteraction)[,-1] # drop intercept

best_lambda = cv.glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=0,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min; best_lambda
```

```
## [1] 19.95262
```

```
ridges.fullinteraction = glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=0,
                      lambda=best_lambda)
imp <- as.data.frame(varImp(ridges.fullinteraction, lambda=best_lambda))
imp <- data.frame(overall = imp$Overall,
           names = rownames(imp))
imp[order(imp$overall,decreasing = T),][1:10,]
```

```
##      overall    names
## 52  6.385091     Fld.
## 607 4.054872   BA:OBP
## 689 3.740522 OBP:Fld.
## 651 3.061896  OBP:SLG
## 13  2.611850      OBP
## 608 2.189910   BA:SLG
## 652 1.919731  OBP:OPS
## 646 1.681135  BA:Fld.
## 731 1.495566 SLG:Fld.
## 609 1.373677   BA:OPS
```

```
X.fullinteraction.test = model.matrix(lm.fullinteraction, data=test.df)[,-1] # drop intercept

yhats.fullinteraction.train = predict(ridges.fullinteraction, X.fullinteraction)
ridgesfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, yhats.fullinteraction.train) # train RM
ridgesfullinteraction.trainR2 = R2(train.df$W.L..next_year, yhats.fullinteraction.train) # train R2

yhats.fullinteraction.test = predict(ridges.fullinteraction, X.fullinteraction.test)
#plot(RMSE.ridges.fullinteraction.test~log(ridges.fullinteraction$lambda, 10), type='l')
ridgesfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, yhats.fullinteraction.test) # train RMSE
ridgesfullinteraction.testR2 = R2(test.df$W.L..next_year, yhats.fullinteraction.test) # train R2
```

```
set.seed(139)
# Lasso Regression
# regularize full model

best_lambda = cv.glmnet(X.full, train.df$W.L..next_year, alpha=1,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min; best_lambda
```

```
## [1] 0.1
```

```
lassos.full = glmnet(X.full, train.df$W.L..next_year, alpha=1,
                      lambda=best_lambda)
imp <- as.data.frame(varImp(lassos.full, lambda=best_lambda))
imp <- data.frame(overall = imp$Overall,
           names = rownames(imp))
imp[order(imp$overall,decreasing = T),][1:10,]
```

```
##       overall names
## 13 51.3466089   OBP
## 14 15.7472823   SLG
## 39  9.1180314  WHIP
## 52  6.1078745  Fld.
```

```
## 26  2.9006011   SHO
## 35  2.1270898    BK
## 54  1.7239656  RF.G
## 40  1.4583395    H9
## 6   1.0723492   X3B
## 41  0.9962776   HR9
```

```
yhats.full.train = predict(lassos.full, X.full)
lassosfull.trainRMSE = RMSE(train.df$W.L..next_year, yhats.full.train) # train RMSE
lassosfull.trainR2 = R2(train.df$W.L..next_year, yhats.full.train) # train R2

yhats.full.test = predict(lassos.full, X.full.test)
#plot(RMSE.lassos.full.test~log(ridges.full$lambda, 10), type='l')
lassosfull.testRMSE = RMSE(test.df$W.L..next_year, yhats.full.test) # test RMSE
lassosfull.testR2 = R2(test.df$W.L..next_year, yhats.full.test) # test RMSE
```

```
set.seed(139)
# regularize full interaction model

best_lambda = cv.glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=1,
                        lambda=10^seq(-4, 4, 0.1))$lambda.min; best_lambda
```

```
## [1] 0.1258925
```

```
lassos.fullinteraction = glmnet(X.fullinteraction, train.df$W.L..next_year, alpha=1,
                       lambda=best_lambda)
imp <- as.data.frame(varImp(lassos.fullinteraction, lambda=best_lambda))
imp <- data.frame(overall = imp$Overall,
           names = rownames(imp))
imp[order(imp$overall,decreasing = T),][1:10,]
```

```
##         overall         names
## 689  44.8055703      OBP:Fld.
## 731  12.6440968      SLG:Fld.
## 13    7.7705702           OBP
## 936   1.3286793        SH:SHO
## 1299  0.7980187  IBB.pitch:BK
## 986   0.7483075        SF:HR9
## 660   0.5828154 OBP:Age.pitch
## 1447  0.3713172      WHIP:SO9
## 1458  0.2711806     WHIP:RF.G
## 349   0.2064814 X3B:HBP.pitch
```

```
yhats.fullinteraction.train = predict(lassos.fullinteraction, X.fullinteraction)
lassosfullinteraction.trainRMSE = RMSE(train.df$W.L..next_year, yhats.fullinteraction.train) # train RM
lassosfullinteraction.trainR2 = R2(train.df$W.L..next_year, yhats.fullinteraction.train) # train R2

yhats.fullinteraction.test = predict(lassos.fullinteraction, X.fullinteraction.test)
#plot(RMSE.lassos.fullinteraction.test~log(lassos.fullinteraction$lambda, 10), type='l')
lassosfullinteraction.testRMSE = RMSE(test.df$W.L..next_year, yhats.fullinteraction.test) # train RMSE
lassosfullinteraction.testR2 = R2(test.df$W.L..next_year, yhats.fullinteraction.test) # train R2
```

```r
set.seed(139)
# Stepwise
lm.step = step(lm.full, scope=c(lower=formula(W.L..next_year~1),
                                upper=lm.fullinteraction), trace=0, direction="both")
formula(lm.step)
```

```
## W.L..next_year ~ Age.bat + PA + AB + H.bat + X3B + OBP + SLG +
##     OPSplus + GDP + HBP.bat + SF + Age.pitch + SV + IP + BK +
##     BF + FIP + WHIP + BB9 + Inn + Ch + PO + A + tot_fa_war3 +
##     num_fas
```

```r
imp <- as.data.frame(varImp(lm.step))
imp <- data.frame(overall = imp$Overall,
           names = rownames(imp))
imp[order(imp$overall,decreasing = T),][1:10,]
```

```
##     overall       names
## 24 4.811593 tot_fa_war3
## 3  4.293909          AB
## 2  3.670723          PA
## 25 3.345379     num_fas
## 5  3.308352         X3B
## 4  3.246229       H.bat
## 18 2.859374        WHIP
## 22 2.759578          PO
## 23 2.718017           A
## 21 2.704481          Ch
```

```r
lmstep.trainRMSE = RMSE(train.df$W.L..next_year, predict(lm.step, newdata=train.df))
lmstep.testRMSE = RMSE(test.df$W.L..next_year, predict(lm.step, newdata=test.df))
lmstep.trainR2 = R2(train.df$W.L..next_year, predict(lm.step, newdata=train.df))
lmstep.testR2 = R2(test.df$W.L..next_year, predict(lm.step, newdata=test.df))
```

```r
set.seed(139)
# model comparison
RMSE.df = data.frame(trainRMSE = c(baseline.trainRMSE,
                                   lmfull.trainRMSE,
                                   lmfullinteraction.trainRMSE,
                                   ridgesfull.trainRMSE,
                                   ridgesfullinteraction.trainRMSE,
                                   lassosfull.trainRMSE,
                                   lassosfullinteraction.trainRMSE,
                                   lmstep.trainRMSE),
                     testRMSE = c(baseline.testRMSE,
                                   lmfull.testRMSE,
                                   lmfullinteraction.testRMSE,
                                   ridgesfull.testRMSE,
                                   ridgesfullinteraction.testRMSE,
                                   lassosfull.testRMSE,
                                   lassosfullinteraction.testRMSE,
                                   lmstep.testRMSE),
                     trainR2 = c(baseline.trainR2,
```

```
                                   lmfull.trainR2,
                                   lmfullinteraction.trainR2,
                                   ridgesfull.trainR2,
                                   ridgesfullinteraction.trainR2,
                                   lassosfull.trainR2,
                                   lassosfullinteraction.trainR2,
                                   lmstep.trainR2),
                     testR2 = c(baseline.testR2,
                                   lmfull.testR2,
                                   lmfullinteraction.testR2,
                                   ridgesfull.testR2,
                                   ridgesfullinteraction.testR2,
                                   lassosfull.testR2,
                                   lassosfullinteraction.testR2,
                                   lmstep.testR2))
rownames(RMSE.df) <- c("baseline", "full", "full interaction",
                       "ridge full", "ridge full interaction",
                       "lasso full", "lasso full interaction",
                       "step")
RMSE.df
```

```
##                           trainRMSE   testRMSE   trainR2       testR2
## baseline               6.273863e+00   7.058916 0.2932730    0.2071767
## full                   5.669061e+00   7.188524 0.4229627    0.1777956
## full interaction       6.104278e-09 157.610140 1.0000000 -394.2470839
## ridge full             5.796197e+00   6.929355 0.3967910    0.2360129
## ridge full interaction 5.741617e+00   6.961877 0.4080977    0.2288248
## lasso full             5.799022e+00   6.948096 0.3962028    0.2318749
## lasso full interaction 5.657099e+00   6.984586 0.4253954    0.2237857
## step                   5.728989e+00   7.082100 0.4106984    0.2019603
```

Decision Tree/Random Forest

```
set.seed(139)
library(rpart)

RMSE = function(y,yhat){
  return(sqrt(mean((y-yhat)^2)))
}

test.df = subset(test.df, test.df$Tm != 'CLE')
tree1 = rpart(formula(lm.full),data=train.df, control = list(minsplit=1,cp=0,maxdepth=20))
yhat.tree1.train = predict(tree1)
yhat.tree1.test = predict(tree1, newdata = test.df)
RMSE.tree1.train = RMSE(train.df$W.L..next_year,yhat.tree1.train)
RMSE.tree1.test = RMSE(test.df$W.L..next_year,yhat.tree1.test)
data.frame(train=RMSE.tree1.train,test=RMSE.tree1.test)
```

```
##    train    test
## 1 3.9511 8.43241
```

```r
set.seed(139)
best.cp = tree1$cptable[,"CP"][which.min(tree1$cptable[,"xerror"])]
tree2 = prune(tree1,best.cp)
yhat.tree2.train = predict(tree2)
yhat.tree2.test = predict(tree2,newdata=test.df)
RMSE.tree2.train = RMSE(train.df$W.L..next_year,yhat.tree2.train)
RMSE.tree2.test = RMSE(test.df$W.L..next_year,yhat.tree2.test)
data.frame(train=RMSE.tree2.train,test=RMSE.tree2.test)
```

```
##      train     test
## 1 6.461346 7.490019
```

```r
set.seed(139)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
maxnodes = c(100,200,500)
ntree= 200
rmses.bag = rep(NA,length(maxnodes))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:length(maxnodes)){
  bagtemp = randomForest(formula(lm.full),data=train.df,
                         mtry=56, maxnodes=maxnodes[i], ntree=ntree)
  rmses.bag[i]=RMSE(train.df$W.L..next_year, bagtemp$predicted)
  if(rmses.bag[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=rmses.bag[i]
    bag=bagtemp
  }
}
data.frame(maxnodes=maxnodes, RMSE=rmses.bag)
```

```
##   maxnodes     RMSE
## 1      100 6.397272
## 2      200 6.460016
## 3      500 6.439697
```

```r
yhat.bag.train = predict(bag)
yhat.bag.test = predict(bag, newdata = test.df)
RMSE.bag.train = RMSE(train.df$W.L..next_year,yhat.bag.train)
RMSE.bag.test = RMSE(test.df$W.L..next_year,yhat.bag.test)
data.frame(train=RMSE.bag.train,test=RMSE.bag.test)
```

```
##      train     test
## 1 6.397272 7.109748
```

```r
library(randomForest)
set.seed(139)
maxnodes = c(100,200,500)
mtry = c(15, 25, 35, 45, 55)
ntree=200
pars = expand.grid(maxnodes=maxnodes,mtry=mtry)
RMSEs = rep(NA,nrow(pars))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:nrow(pars)){
  rftemp = randomForest(formula(lm.full),data=train.df,
                        mtry=pars$mtry[i], maxnodes=pars$maxnodes[i], ntree=ntree)
  RMSEs[i]=RMSE(train.df$W.L..next_year, rftemp$predicted)
  if(RMSEs[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=RMSEs[i]
    rf1=rftemp
  }
}
data.frame(maxnodes=pars$maxnodes,mtry=pars$mtry,RMSE=RMSEs)
```

```
##    maxnodes mtry     RMSE
## 1       100   15 6.434901
## 2       200   15 6.476828
## 3       500   15 6.464521
## 4       100   25 6.460300
## 5       200   25 6.447135
## 6       500   25 6.471577
## 7       100   35 6.417410
## 8       200   35 6.416950
## 9       500   35 6.431414
## 10      100   45 6.505922
## 11      200   45 6.477685
## 12      500   45 6.377898
## 13      100   55 6.488894
## 14      200   55 6.483632
## 15      500   55 6.434556
```

```r
pars[which(RMSEs==bestRMSE),]
```

```
##    maxnodes mtry
## 12      500   45
```

```r
yhat.rf1.train = predict(rf1)
yhat.rf1.test = predict(rf1, newdata = test.df)
RMSE.rf1.train = RMSE(train.df$W.L..next_year,yhat.rf1.train)
RMSE.rf1.test = RMSE(test.df$W.L..next_year,yhat.rf1.test)
data.frame(train=RMSE.rf1.train,test=RMSE.rf1.test)
```

```
##      train     test
## 1 6.377898 7.137898
```

```r
importance(rf1)
```

```
##                 IncNodePurity
## Age.bat              379.6500
## PA                   134.2052
## AB                   202.4752
## H.bat                153.2252
## X2B                  235.4155
## X3B                  440.7061
## HR.bat               662.0776
## SB                   329.2822
## CS                   421.0483
## BB.bat               829.3347
## SO.bat               406.1870
## BA                   295.5659
## OBP                  546.6960
## SLG                  430.7280
## OPS                  484.9618
## OPSplus              413.0187
## TB                   152.3767
## GDP                  432.1897
## HBP.bat              460.4134
## SH                   337.8594
## SF                   268.6063
## IBB.bat              498.5717
## Age.pitch            743.6306
## W.L..same_year      3150.8236
## GF                   300.0394
## SHO                  290.4883
## SV                  1258.3371
## IP                   293.3168
## H.pitch              272.8069
## HR.pitch             296.6186
## BB.pitch             348.5045
## IBB.pitch            271.5517
## SO.pitch             536.0281
## HBP.pitch            291.8748
## BK                   454.3285
## WP                   381.2293
## BF                   218.8919
## FIP                  740.6101
## WHIP                2256.6089
## H9                  1279.5906
```

```
## HR9                   397.9306
## BB9                   609.8318
## SO9                   325.4082
## SO.W                  454.5702
## G                     483.0792
## Inn                   253.6438
## Ch                    248.2340
## PO                    342.0238
## A                     328.3512
## E                     523.4479
## DP                    406.5807
## Fld.                  457.3072
## RF.9                  421.2885
## RF.G                  462.9082
## tot_fa_war3          1220.7219
## num_fas               481.4762
```

```
library(randomForest)
set.seed(139)
maxnodes = c(100,200,500)
mtry = c(1,3,5)
ntree=200
pars = expand.grid(maxnodes=maxnodes,mtry=mtry)
RMSEs = rep(NA,nrow(pars))
bestRMSE = sd(train.df$W.L..next_year)

for(i in 1:nrow(pars)){
  rftemp = randomForest(W.L..next_year ~ W.L..same_year + WHIP + H9 + SV + tot_fa_war3, data=train.df,
                        mtry=pars$mtry[i], maxnodes=pars$maxnodes[i], ntree=ntree)
  RMSEs[i]=RMSE(train.df$W.L..next_year, rftemp$predicted)
  if(RMSEs[i]<bestRMSE){
    best_maxnodes = maxnodes[i]
    bestRMSE=RMSEs[i]
    rf2=rftemp
  }
}
data.frame(maxnodes=pars$maxnodes,mtry=pars$mtry,RMSE=RMSEs)
```

```
##   maxnodes mtry     RMSE
## 1      100    1 6.498728
## 2      200    1 6.512211
## 3      500    1 6.500416
## 4      100    3 6.469136
## 5      200    3 6.513616
## 6      500    3 6.453573
## 7      100    5 6.462556
## 8      200    5 6.461979
## 9      500    5 6.496799
```

```
pars[which(RMSEs==bestRMSE),]
```

```
##   maxnodes mtry
## 6      500    3
```

```r
yhat.rf2.train = predict(rf2)
yhat.rf2.test = predict(rf2, newdata = test.df)
RMSE.rf2.train = RMSE(train.df$W.L..next_year,yhat.rf2.train)
RMSE.rf2.test = RMSE(test.df$W.L..next_year,yhat.rf2.test)
data.frame(train=RMSE.rf2.train,test=RMSE.rf2.test)
```

```
##      train     test
## 1 6.453573 7.706507
```

```r
importance(rf2)
```

```
##                IncNodePurity
## W.L..same_year      6705.507
## WHIP                6323.767
## H9                  4838.534
## SV                  5495.341
## tot_fa_war3         4804.368
```

```r
set.seed(139)
tree3 = rpart(W.L..next_year ~ W.L..same_year + WHIP + H9 + SV + tot_fa_war3,
              data=train.df, control = list(minsplit=1, cp=0, maxdepth=20))
yhat.tree3.train = predict(tree3)
yhat.tree3.test = predict(tree3, newdata = test.df)
RMSE.tree3.train = RMSE(train.df$W.L..next_year,yhat.tree3.train)
RMSE.tree3.test = RMSE(test.df$W.L..next_year,yhat.tree3.test)
data.frame(train=RMSE.tree3.train,test=RMSE.tree3.test)
```

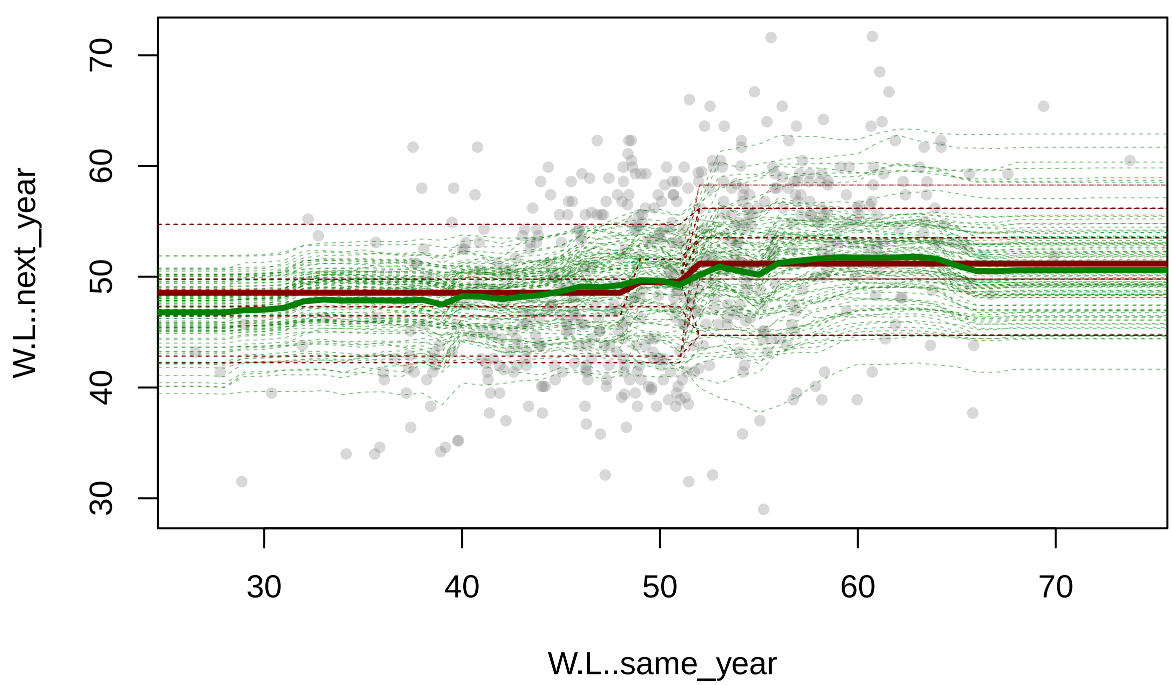
```
##      train     test
## 1 4.842705 8.420162
```

```r
set.seed(139)
best.cp = tree3$cptable[,"CP"][which.min(tree3$cptable[,"xerror"])]
tree4 = prune(tree3,best.cp)
yhat.tree4.train = predict(tree4)
yhat.tree4.test = predict(tree4,newdata=test.df)
RMSE.tree4.train = RMSE(train.df$W.L..next_year,yhat.tree4.train)
RMSE.tree4.test = RMSE(test.df$W.L..next_year,yhat.tree4.test)
data.frame(train=RMSE.tree4.train,test=RMSE.tree4.test)
```

```
##      train     test
## 1 6.010486 8.204083
```

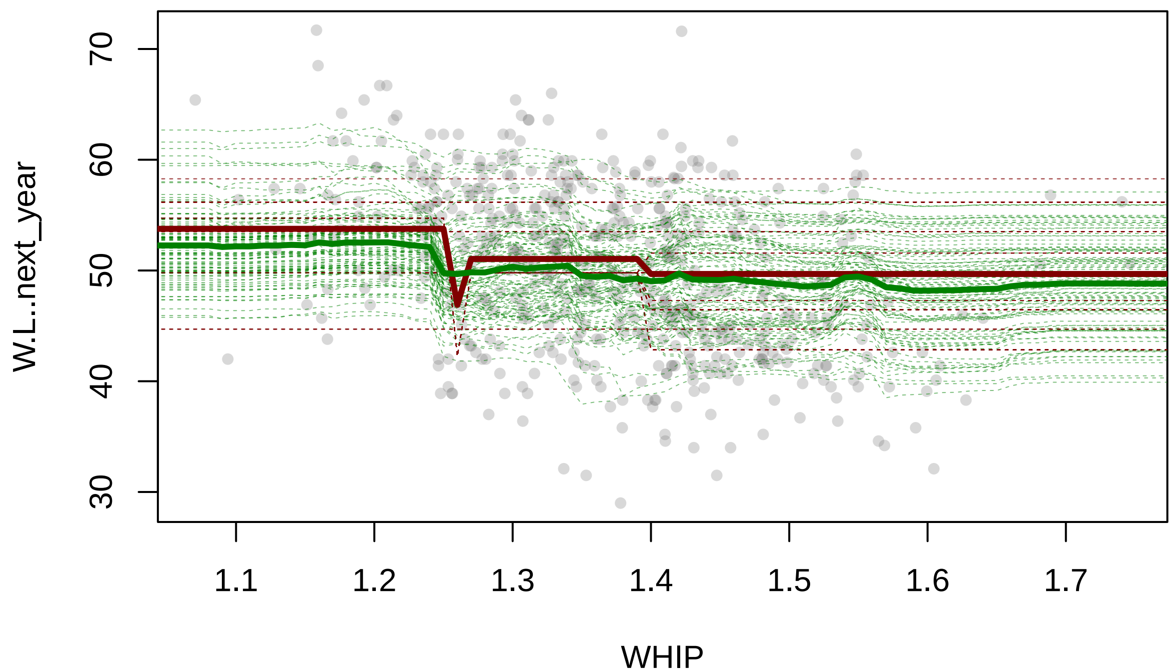
```r
library(randomForest)
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(0,100,1)
plot(W.L..next_year~W.L..same_year, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
```

```
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$W.L..same_year=dummyx
  yhat = predict(tree4,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf2,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



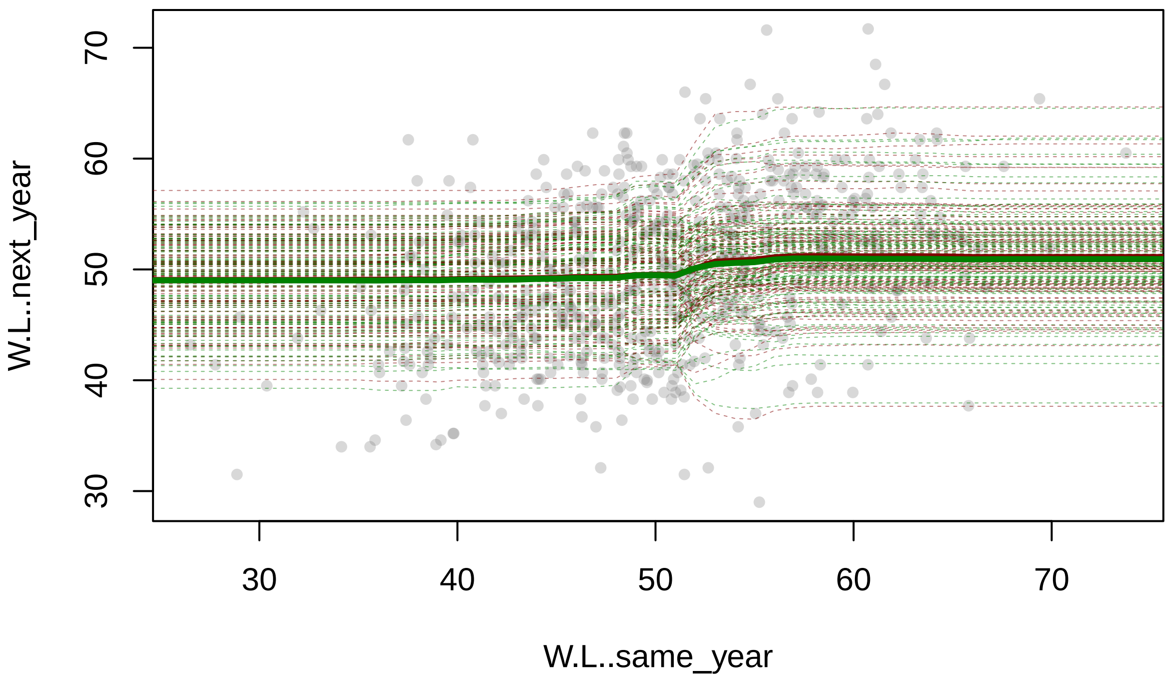

```
library(randomForest)
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(1,2,.01)
plot(W.L..next_year~WHIP, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
```

```
  rows$WHIP=dummyx
  yhat = predict(tree4,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf2,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



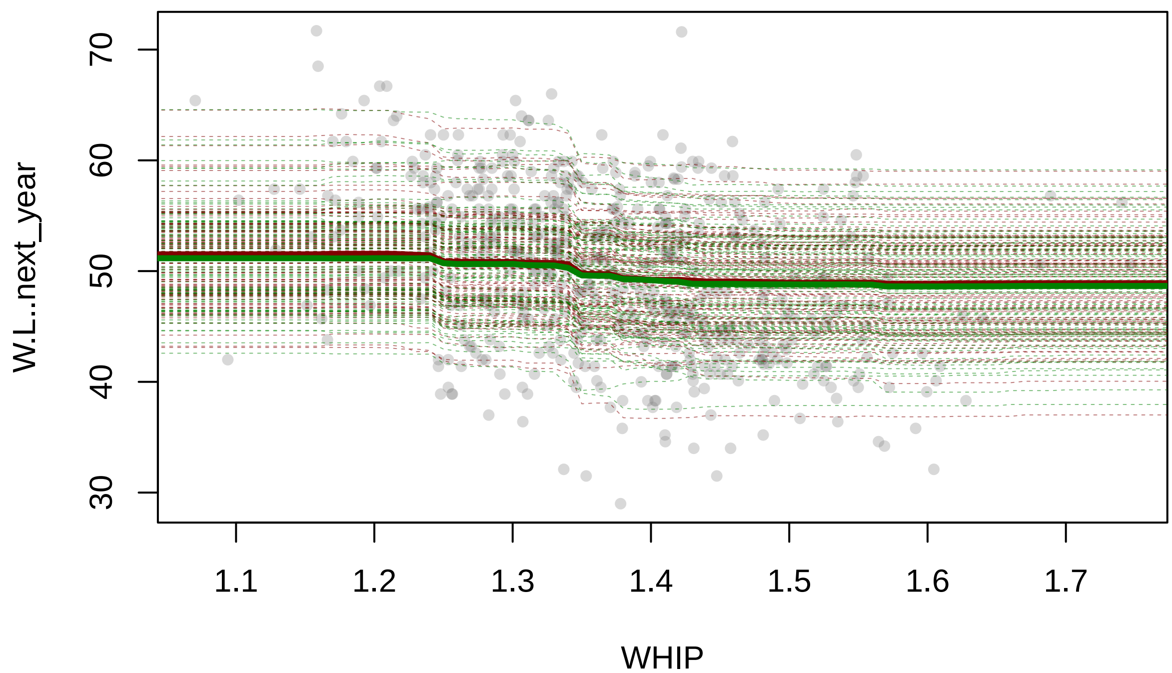

```
library(randomForest)
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(0,100,1)
plot(W.L..next_year~W.L..same_year, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$W.L..same_year=dummyx
  yhat = predict(bag,new=rows)
```

```
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
  yhat.rf = predict(rf1,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```

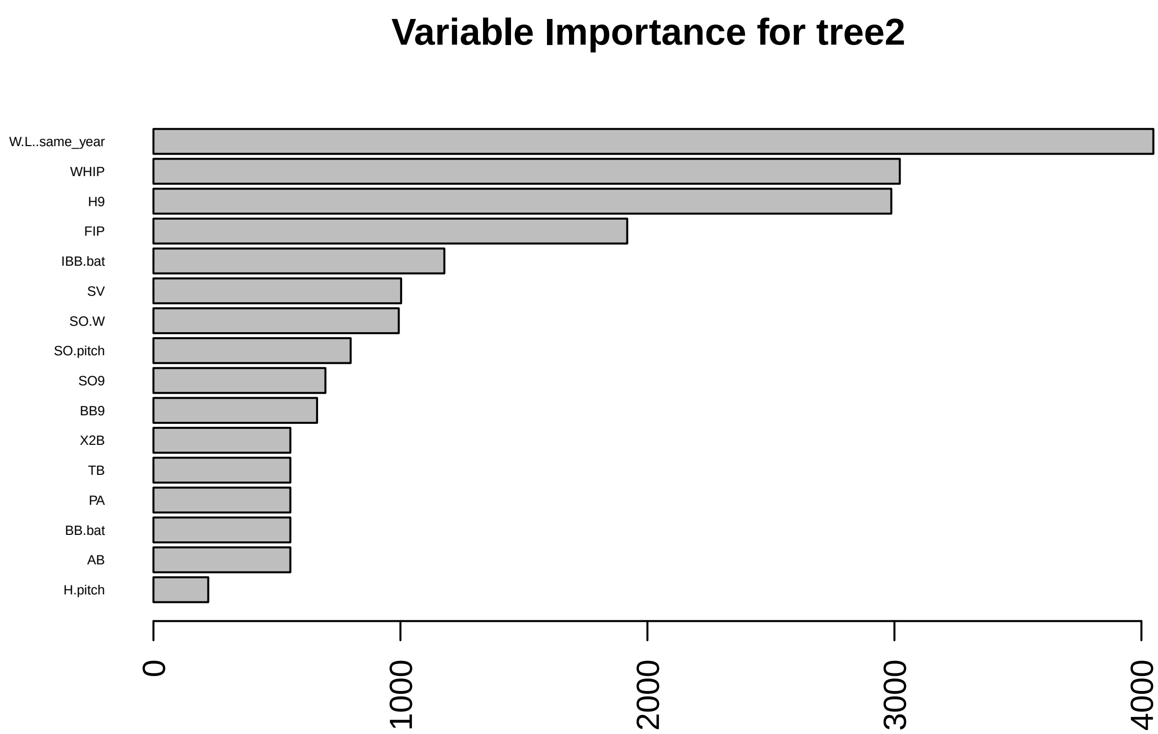



```
library(randomForest)
set.seed(139)
samp = sample(nrow(train.df),100)
dummy_df = train.df[samp,]
dummyx = seq(1,2,.01)
plot(W.L..next_year~WHIP, data=train.df,cex=0.8,pch=16,col=rgb(0.5,0.5,0.5,0.3))
yhats = matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
yhats.rf=matrix(NA,nrow=nrow(dummy_df),ncol=length(dummyx))
for(i in 1:nrow(dummy_df)){
  rows=dummy_df[rep(i,length(dummyx)),]
  rows$WHIP=dummyx
  yhat = predict(bag,new=rows)
  lines(yhat~dummyx,col=rgb(0.5,0,0,0.5),lwd=0.5,lty=2:3)
  yhats[i,]=yhat
```

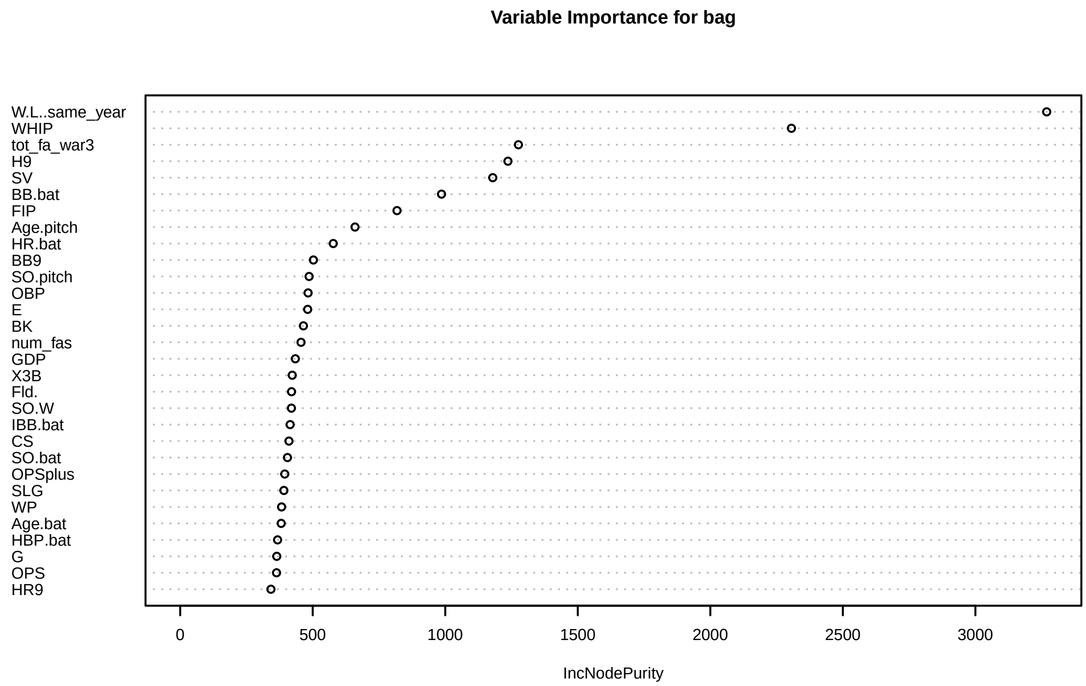
```
  yhat.rf = predict(rf1,new=rows)
  lines(yhat.rf~dummyx,col=rgb(0,0.5,0,0.5),lwd=0.5,lty=2:3)
  yhats.rf[i,]=yhat.rf
}
mean_yhat = apply(yhats,2,mean)
mean_yhat.rf = apply(yhats.rf,2,mean)
lines(mean_yhat~dummyx,col=rgb(0.5,0,0,1),lwd=3)
lines(mean_yhat.rf~dummyx,col=rgb(0,0.5,0,1),lwd=3)
```



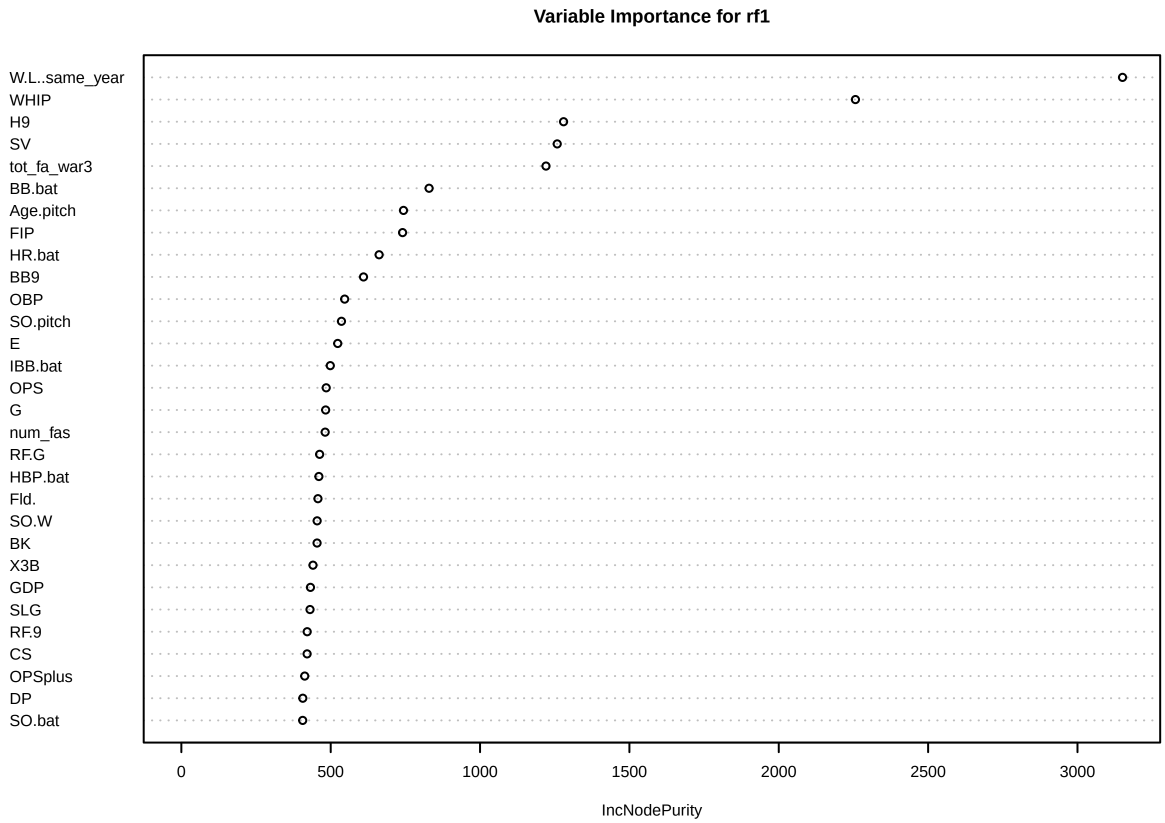

```
set.seed(139)
barplot(sort(tree2$variable.importance),horiz = T,las=2,cex.names = 0.4, main='Variable Importance for t
```



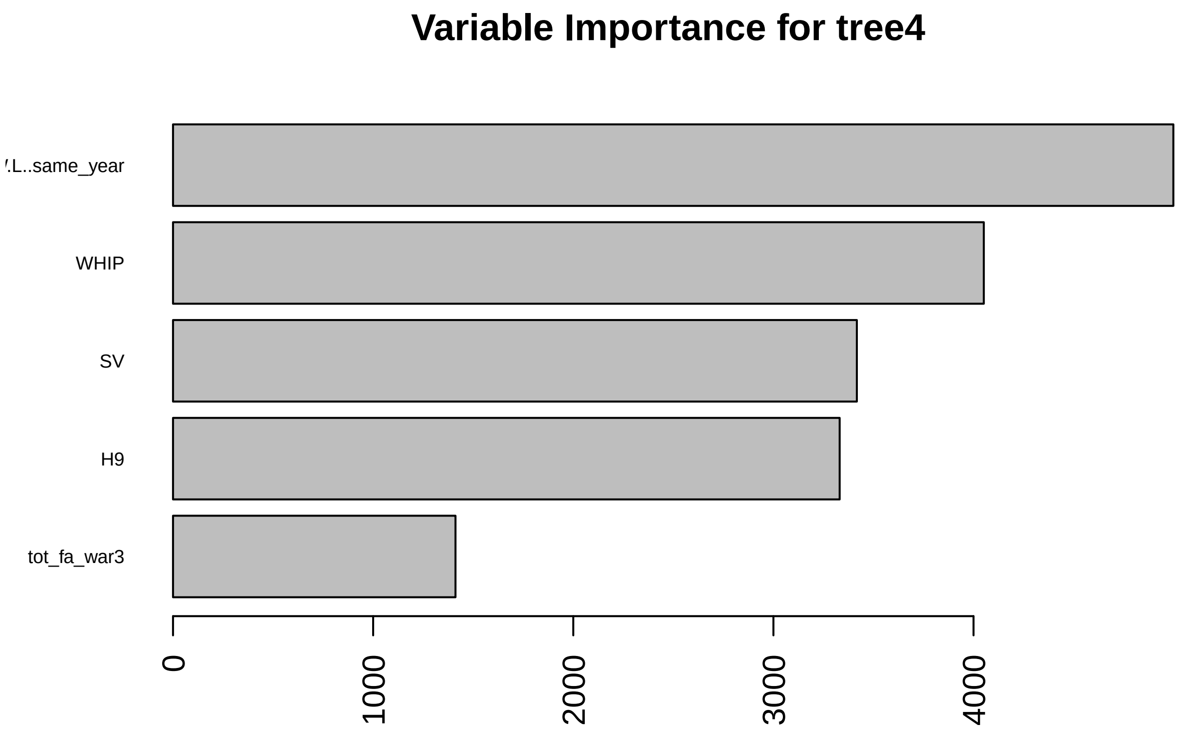

```
varImpPlot(bag, cex=0.5, main='Variable Importance for bag')
```



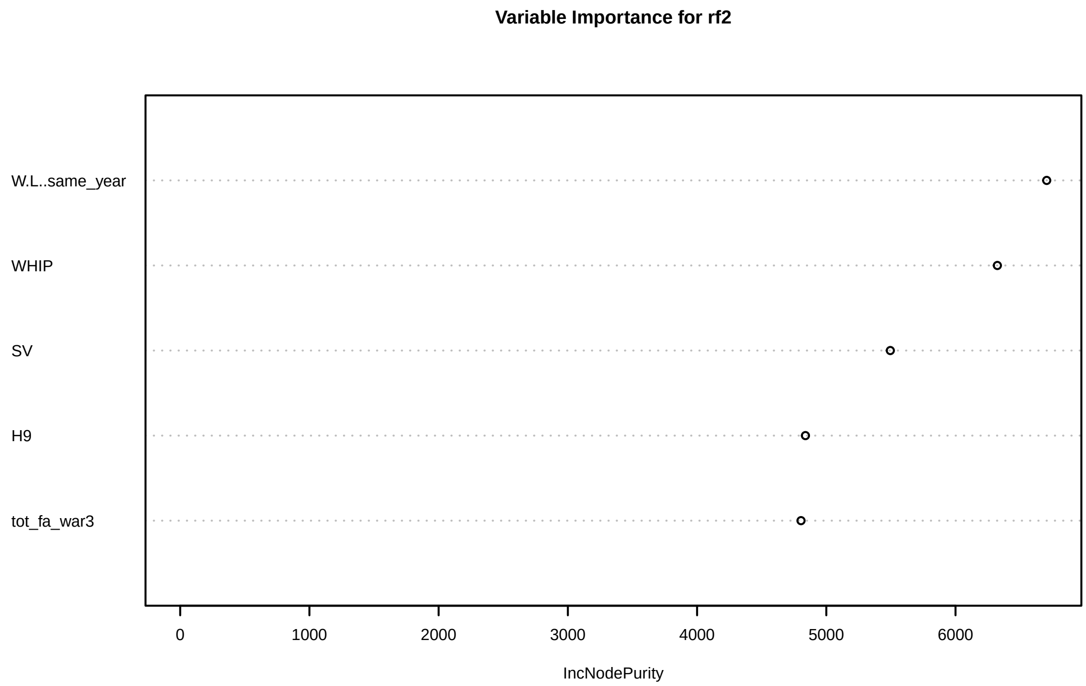

```
varImpPlot(rf1,cex=0.5, main='Variable Importance for rf1')
```




```
barplot(sort(tree4$variable.importance),horiz = T,las=2,cex.names = 0.6, main='Variable Importance for t
```




```
varImpPlot(rf2, cex=0.5, main='Variable Importance for rf2')
```




```
set.seed(139)
tab <- matrix(c(RMSE.tree1.train, RMSE.tree1.test,
  RMSE.tree2.train, RMSE.tree2.test,
  RMSE.bag.train, RMSE.bag.test,
  RMSE.rf1.train, RMSE.rf1.test,
  RMSE.rf2.train, RMSE.rf2.test,
  RMSE.tree4.train, RMSE.tree4.test), nrow=6, byrow = TRUE
)
colnames(tab) <- c('train','test')
rownames(tab) <- c('tree1','tree2','bag', 'rf1', 'rf2', 'tree4')
tab <- as.table(tab)
tab
```

```
##          train     test
## tree1 3.951100 8.432410
## tree2 6.461346 7.490019
## bag   6.397272 7.109748
## rf1   6.377898 7.137898
## rf2   6.453573 7.706507
## tree4 6.010486 8.204083
```

Mixed Effects Models

```r
set.seed(139)
library(lme4)


# for (i in 1997:2022){
#   lmer_model <- lmer(team_data[[i]]$W.L.~poly(team_data[[i]]$BatAge, 2, raw = TRUE) + (1 + poly(team_
#   summary(lmer_model)
# }

lmer_model <- lmer(train.df$W.L..next_year ~ poly(train.df$Age.bat, 2, raw = FALSE) + (1 + poly(train.d
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$Age.bat, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$Age.bat, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3574.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8353 -0.7280  0.0241  0.6893  3.3827
##
## Random effects:
##  Groups        Name                                    Variance Std.Dev. Corr
##  train.df.Tm   (Intercept)                              13.71    3.703
##  train.df.Tm.1 poly(train.df$Age.bat, 2, raw = FALSE)1 231.90   15.228
##                poly(train.df$Age.bat, 2, raw = FALSE)2 238.85   15.455   -1.00
##  Residual                                               42.18    6.495
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                         Estimate Std. Error t value
## (Intercept)                              49.8660     0.7485  66.623
## poly(train.df$Age.bat, 2, raw = FALSE)1  -5.3344     8.0431  -0.663
## poly(train.df$Age.bat, 2, raw = FALSE)2   6.7511     7.9705   0.847
##
## Correlation of Fixed Effects:
##                    (Intr) p(.$A.,2,r=FALSE)1
## p(.$A.,2,r=FALSE)1 -0.009
## p(.$A.,2,r=FALSE)2  0.023 -0.132
```

```r
lmer_model <- lmer(train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) + (1 + poly(train.df$BA,
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$BA, 2, raw = FALSE) |
```

```
##         train.df$Tm))
##
## REML criterion at convergence: 3562.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8878 -0.6824  0.0135  0.6932  3.2131
##
## Random effects:
##  Groups        Name                             Variance  Std.Dev. Corr
##  train.df.Tm   (Intercept)                      13.870992 3.72438
##  train.df.Tm.1 poly(train.df$BA, 2, raw = FALSE)1  1.522582 1.23393
##                poly(train.df$BA, 2, raw = FALSE)2  0.004076 0.06384  1.00
##  Residual                                       41.796196 6.46500
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                    Estimate Std. Error t value
## (Intercept)                         49.9374     0.7497  66.612
## poly(train.df$BA, 2, raw = FALSE)1  33.4075     8.7293   3.827
## poly(train.df$BA, 2, raw = FALSE)2   3.5883     6.8383   0.525
##
## Correlation of Fixed Effects:
##                    (Intr) p(.$BA,2,r=FALSE)1
## p(.$BA,2,r=FALSE)1 0.018
## p(.$BA,2,r=FALSE)2 0.005  0.027
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
# lmer_model <- lmer(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B +
                # HR.bat + SB + CS + BB.bat + SO.bat + BA + OBP + SLG + OPS + OPSplus +
                # TB + GDP + HBP.bat + SH + SF + IBB.bat + Age.pitch + W.L..same_year +
                # GF + SHO + SV + IP + H.pitch + HR.pitch +
                # BB.pitch + IBB.pitch + SO.pitch + HBP.pitch + BK + WP + BF +
                # FIP + WHIP + H9 + HR9 + BB9 + SO9 + SO.W +
                # G + Inn + Ch + PO + A + E + DP + Fld. +
                # RF.9 + RF.G + tot_fa_war3 + num_fas || Tm, data = train.df, verbose=TRUE)
```

```
set.seed(139)
summary(lmer_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: train.df$W.L..next_year ~ poly(train.df$BA, 2, raw = FALSE) +
##     ((1 | train.df$Tm) + (0 + poly(train.df$BA, 2, raw = FALSE) |
##         train.df$Tm))
##
## REML criterion at convergence: 3562.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8878 -0.6824  0.0135  0.6932  3.2131
##
## Random effects:
```

```
##  Groups        Name                              Variance  Std.Dev. Corr
##  train.df.Tm   (Intercept)                       13.870992 3.72438
##  train.df.Tm.1 poly(train.df$BA, 2, raw = FALSE)1  1.522582 1.23393
##                poly(train.df$BA, 2, raw = FALSE)2  0.004076 0.06384  1.00
##  Residual                                        41.796196 6.46500
## Number of obs: 536, groups:  train.df$Tm, 29
##
## Fixed effects:
##                                   Estimate Std. Error t value
## (Intercept)                        49.9374     0.7497  66.612
## poly(train.df$BA, 2, raw = FALSE)1  33.4075     8.7293   3.827
## poly(train.df$BA, 2, raw = FALSE)2   3.5883     6.8383   0.525
##
## Correlation of Fixed Effects:
##                    (Intr) p(.$BA,2,r=FALSE)1
## p(.$BA,2,r=FALSE)1 0.018
## p(.$BA,2,r=FALSE)2 0.005  0.027
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + (1 + WHIP + W.L..same_year +
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.007809
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.051978
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch | Tm, data = train.df)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.860842
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.997159
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + tot_fa_war3 | Tm, data = trai
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.839756
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.029939
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + H9 | Tm, data = train.df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.631575
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.984588
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + H9 + (1 + WHIP + W.L..same_y
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.396988 (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unide
##  - Rescale variables?;Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
```

```r
# summary(lmer.varmodel)
# predict(lmer.varmodel)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.687929
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.944894
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB | Tm, data = train.df)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.957596
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.091514
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB + (1 + Age.bat + PA + AB | Tm) , data = train.
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.946579
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.091969
```

RIDGE full

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ OBP + Fld. + SLG + OPS + WHIP + (1 + OBP + Fld. + SLG + OPS + WH
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.489772
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.966577
```

```
set.seed(139)
# lmer.varmodel <- lmer(W.L..next_year ~ OBP + SLG + OPS + SHO + Fld. + (1 + OBP + SLG + OPS + SHO + Fl
# RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
# RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ OBP + Fld. + SLG + OPS + WHIP + (1 | Tm), data = train.df)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.946799
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.929084
```

RIDGE full interaction

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Fld. + BA:OBP + OBP:Fld. + OBP:SLG + OBP + (1 + Fld. + BA:OBP + (
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.92393
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.081875
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ Fld. + BA:OBP + OBP:Fld. + OBP:SLG + OBP + (1 | Tm) , data = trai
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.212734
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.018323
```

LASSO full

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ OBP + SLG + WHIP + Fld. + SHO + (1 + OBP + SLG + WHIP + Fld. + SH
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.435744
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.938336
```

```r
summary(lmer.varmodel)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: W.L..next_year ~ OBP + SLG + WHIP + Fld. + SHO + (1 + OBP + SLG +
##     WHIP + Fld. + SHO | Tm)
##    Data: train.df
##
## REML criterion at convergence: 3437.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.4227 -0.5665  0.0229  0.6163  2.9642
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  Tm       (Intercept) 25623.90 160.075
##           OBP          1484.35  38.527   0.10
##           SLG           649.72  25.490  -0.36 -0.77
##           WHIP          259.12  16.097  -0.38  0.07  0.58
##           Fld.        24288.41 155.847  -0.98 -0.15  0.31  0.22
##           SHO            43.69   6.610   0.56  0.20 -0.78 -0.94 -0.43
##  Residual                32.82   5.729
## Number of obs: 536, groups:  Tm, 29
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   41.295     45.904   0.900
## OBP           65.398     27.038   2.419
## SLG           20.537     15.950   1.288
## WHIP         -23.215      4.073  -5.699
## Fld.          12.322     45.713   0.270
```

```
## SHO            6.042      3.242   1.864
##
## Correlation of Fixed Effects:
##      (Intr) OBP    SLG    WHIP   Fld.
## OBP   0.048
## SLG  -0.129 -0.812
## WHIP -0.301 -0.051  0.150
## Fld. -0.986 -0.119  0.131  0.176
## SHO   0.171 -0.200  0.032 -0.241 -0.121
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ OBP + SLG + WHIP + Fld. + SHO + (1 | Tm) , data = train.df)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.932623
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.831064
```

```
summary(lmer.varmodel)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: W.L..next_year ~ OBP + SLG + WHIP + Fld. + SHO + (1 | Tm)
##    Data: train.df
##
## REML criterion at convergence: 3464.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.3770 -0.6269 -0.0280  0.6929  3.1814
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Tm       (Intercept)  7.52    2.742
##  Residual             37.10    6.091
## Number of obs: 536, groups:  Tm, 29
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   33.292     34.791   0.957
## OBP           48.052     27.099   1.773
## SLG           33.469     15.913   2.103
## WHIP         -21.499      2.737  -7.855
## Fld.          18.524     35.116   0.528
## SHO            6.184      3.015   2.051
##
## Correlation of Fixed Effects:
##      (Intr) OBP    SLG    WHIP   Fld.
## OBP   0.031
```

```
## SLG  -0.083 -0.810
## WHIP -0.228 -0.115  0.027
## Fld. -0.987 -0.115  0.098  0.145
## SHO   0.043 -0.231  0.131  0.031 -0.022
```

LASSO full interaction

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ OBP:Fld. + SLG:Fld. + OBP + SH:SHO + IBB.pitch:BK + (1 + OBP:Fld
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00918726 (tol = 0.002, component 1)
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.843151
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 6.928967
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ OBP:Fld. + SLG:Fld. + OBP + SH:SHO + IBB.pitch:BK + (1 | Tm) , da
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.157779
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.061127
```

Step

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ tot_fa_war3 + AB + PA + num_fas + X3B + (1 + tot_fa_war3 + AB + P
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.762591
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.123673
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ tot_fa_war3 + AB + PA + num_fas + X3B + (1 | Tm), data = train.df
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.140747
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.102855
```

Random Forest 1 and Random Forest 2

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + FIP + H9 + SO.W + BB9 + (1 + WHIP + FIP + H9 + SO.W + BB9
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.672768
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.139875
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + FIP + H9 + SO.W + BB9 + (1 | Tm), data = train.df)
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.191988
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.200265
```

Bag

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + tot_fa_war3 + IBB.bat + (1 +
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.904282
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.203873
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + Age.pitch + tot_fa_war3 + IBB.bat + (1 |
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.07536
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.125792
```

rf1

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + H9 + FIP + tot_fa_war3 + (1 + WHIP + W.L
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.75037
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.191054
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ WHIP + W.L..same_year + H9 + FIP + tot_fa_war3 + (1 | Tm), data =
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.093042
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.259996
```

Pruned Decision tree with 3 Predictors

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + Age.pitch | Tm , data = train.df)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.763283
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.052558
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + Age.pitch + (1 + W.L..same_year + WHIP +
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.89997
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.046541
```

Pruned Decision tree with all predictors

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + H9 | Tm , data = train.df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.659651
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.056473
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + H9 + (1 + W.L..same_year + WHIP + H9 | Tr
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.718808
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.07585
```

Testing stuff

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + H9 + FIP + IBB.bat + (1 + W.L..same_year
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.609329
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.006688
```

```
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + H9 + SV + tot_fa_war3 + (1 + W.L..same_y
```

```
## boundary (singular) fit: see help('isSingular')
```

```
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 5.552136
```

```
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.373136
```

```r
set.seed(139)
lmer.varmodel <- lmer(W.L..next_year ~ W.L..same_year + WHIP + H9 + SV + tot_fa_war3 + (1 | Tm) , data =
RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
```

```
## [1] 6.004241
```

```r
RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```

```
## [1] 7.394825
```

```r
set.seed(139)
# lmer.varmodel <- lmer(W.L..next_year ~ Age.bat + PA + AB + H.bat + X2B + X3B + HR.bat + SB + CS + BB.
# RMSE(train.df$W.L..next_year, predict(lmer.varmodel))
# RMSE(test.df$W.L..next_year, predict(lmer.varmodel, newdata=test.df))
```